UBx :: IUT-INFO [R4.A.08]
Introduction à la Virtualisation

Vincent Autefage
vincent.autefage@u-bordeaux.com

université de BORDEAUX

Informatique / iut de BORDEAUX

Organisation du module

Enseignants

- Vincent Autefage [B, C]
- Pierre Ramet [A]

Organisation du module

Enseignants

- Vincent Autefage [B, C]
- Pierre Ramet [A]

Évaluation

- Contrôle continu
- TP noté

Organisation du module

Enseignants

- Vincent Autefage [B, C]
- Pierre Ramet [A]

Évaluation

- Contrôle continu
- TP noté

Planning

R.4.A08 - S4 Introduction et rappels systèmes
R.4.A08 - S4 Virtualisation lourde - VM
R.4.A08 - S4 Virtualisation légère - Conteneur
R.4.A08 - S4 Introduction à l'orchestration des conteneurs

R.5.A09 - S5 Orchestration
R.5.A09 - S5 Sécurité et optimisation des conteneurs
R.5.A09 - S5 Isolation bas niveau
R.5.A09 - S5 Cloud-Computing

Virtualisation - Définition

Virtualisation

La **virtualisation** est une technique permettant d'**exécuter** un ou des **éléments de n'importe quelle couche** (du système d'exploitation au service applicatif en passant par le matériel) de manière **plus ou moins isolée**. Pour ce faire, une **couche d'abstraction** (et éventuellement d'encapsulation) supplémentaire est introduite.

Vincent Autefage Introduction à la Virtualisation 3/18

Virtualisation - Définition

Virtualisation

La **virtualisation** est une technique permettant d'**exécuter** un ou des **éléments de n'importe quelle couche** (du système d'exploitation au service applicatif en passant par le matériel) de manière **plus ou moins isolée**. Pour ce faire, une **couche d'abstraction** (et éventuellement d'encapsulation) supplémentaire est introduite.

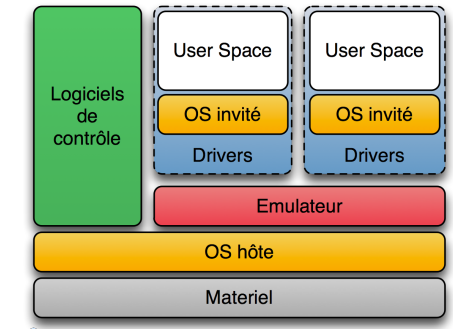
Virtualisation système

La **virtualisation système** est une technique de virtualisation dédiée aux **systèmes d'exploitation**. Elle permet donc d'**utiliser plusieurs systèmes d'exploitation** de manière **plus ou moins isolée** entre eux mais également de l'hôte.

Différents types de virtualisation système

- Émulateur (*Hosted Hypervisor / type 2*)

Source : Wikimedia



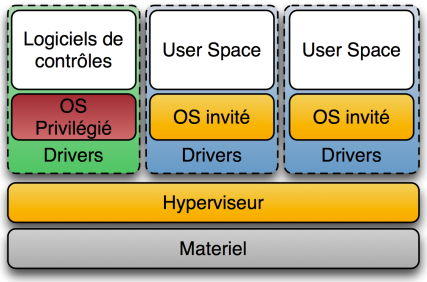
Les **émulateurs** virtualisent **tout ou partie du matériel** dans le but de faire tourner en **espace utilisateur** un système complet incluant **son propre noyau** ainsi que ses **drivers**.

Vincent Autefage Introduction à la Virtualisation 4/18

Différents types de virtualisation système

- Émulateur (*Hosted Hypervisor / type 2*)
- Hyperviseur (*Native Hypervisor / type 1*)

Source : wikimedia

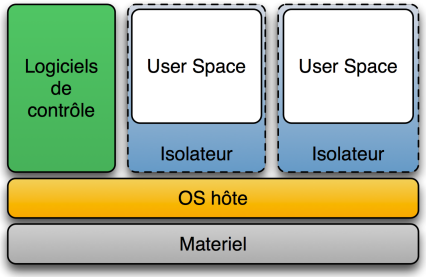


Les **hyperviseurs natifs** s'insèrent **entre le matériel et le noyau** de chaque machine virtuelle ajoutant ainsi une **couche d'abstraction supplémentaire** pour accéder au matériel. On parle plus généralement de **micro-noyau**.

Différents types de virtualisation système

- Émulateur (*Hosted Hypervisor / type 2*)
- Hyperviseur (*Native Hypervisor / type 1*)
- Conteneur (*Container*)

Source : wikimedia



OpenVZ cri-o LXC docker podman

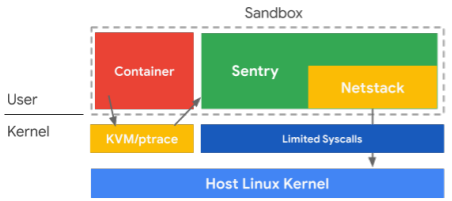
Les **conteneurs** sont centrés sur des **images disques** contenant un **système de fichiers dédié** mais **reposent sur le même noyau** que celui du système hôte. Ils proposent une **isolation multi-niveaux** (*e.g.* systèmes de fichiers, utilisateurs, processus, réseau).

Vincent Autefage Introduction à la Virtualisation 4/18

Différents types de virtualisation système

- Émulateur (*Hosted Hypervisor / type 2*)
- Hyperviseur (*Native Hypervisor / type 1*)
- Conteneur (*Container*)
- Virtualisation légère (*Lightweight Virtualization*)

Source : GVisor



kata containers gVisor Firecracker

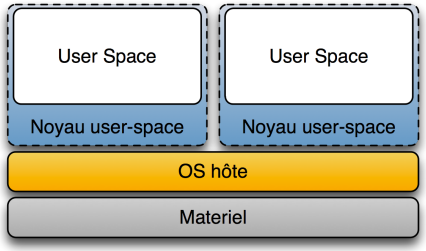
La **virtualisation légère** ou **micro-VM** est à **mi-chemin** entre un **conteneur** et une **machine virtuelle** exploitant un **pseudo-noyau virtualisé** notamment pour la gestion des appels systèmes.

Vincent Autefage Introduction à la Virtualisation 4/18

Différents types de virtualisation système

- Émulateur (*Hosted Hypervisor / type 2*)
- Hyperviseur (*Native Hypervisor / type 1*)
- Conteneur (*Container*)
- Virtualisation légère (*Lightweight Virtualization*)
- Noyau en mode utilisateur (*User Mode Kernel*)

Source : wikimedia



Un **noyau en mode utilisateur** est un noyau **spécifiquement compilé** pour tourner en **espace utilisateur** au moyen d'une interface d'**appels systèmes spécifiques**.

Vincent Autefage Introduction à la Virtualisation 4/18

Différents types de virtualisation système

- Émulateur (*Hosted Hypervisor / type 2*)
- Hyperviseur (*Native Hypervisor / type 1*)
- Conteneur (*Container*)
- Virtualisation légère (*Lightweight Virtualization*)
- Noyau en mode utilisateur (*User Mode Kernel*)
- Accélération matérielle (*Hardware-assisted / Hardware-accelerated*)

Microsoft Hyper-V

KVM

L'**accélération matérielle** permet de tirer partie des **processeurs modernes** en y exécutant **directement** certaines instructions sans passer par la couche matérielle émulée.

Différents types de virtualisation système

- Émulateur (*Hosted Hypervisor / type 2*)
- Hyperviseur (*Native Hypervisor / type 1*)
- Conteneur (*Container*)
- Virtualisation légère (*Lightweight Virtualization*)
- Noyau en mode utilisateur (*User Mode Kernel*)
- Accélération matérielle (*Hardware-assisted / Hardware-accelerated*)
- Para-virtualisation (*Para-virtualization*)

Microsoft Hyper-V

KVM

Xen™

V1

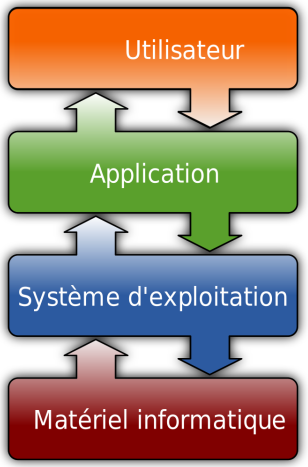
La **para-virtualisation** introduit une **couche logicielle** dont l'**interface** est **équivalente** à celle du **matériel** sans pour autant émuler complémentent ce dernier.

Système d'exploitation et amorçage

Système d'exploitation

- Un **système d'exploitation** (*Operating System*) est un **ensemble de logiciels** destinés à faciliter l'exploitation d'une machine et notamment de ses **ressources matérielles** (*e.g.* processeur, mémoire, stockage, périphériques).
- Il est composé de plusieurs éléments dont un **noyau**, des **pilotes**, un **système de fichiers** et d'une **interface d'appels systèmes** permettant aux **applications** conçus pour ce système de dialoguer avec le noyau.
- Un **pilote** (*Driver*) est un **logiciel** fournissant au système d'exploitation une **interface de dialogue pour un matériel spécifique**. Le pilote peut être vu comme un **traducteur** permettant d'**interpréter les messages du noyau en messages compréhensibles par le matériel et vice-versa**.

Source : wikimedia

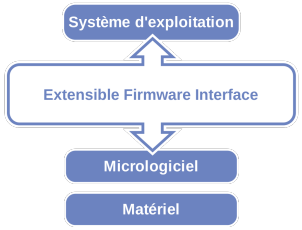


Système d'exploitation et amorçage

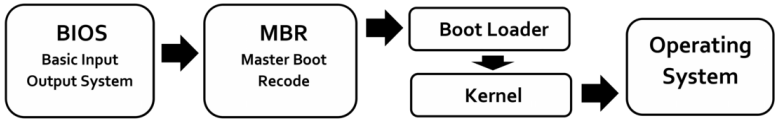
Amorçage

- Un **chargeur d'amorçage** est un **micro-logiciel** (*Firmware*) chargé d'**initialiser le matériel** au démarrage de la machine et de **charger les éléments du système d'exploitation** permettant au noyau de ce dernier de prendre le relais.
- Les deux chargeurs d'amorçage les plus utilisés sont le **BIOS** et l'**UEFI**.
- Sur ces chargeurs doit être installé un **programme d'amorçage** (*Bootloader*) dont les plus courants sont **GRUB** pour Linux, **BootCamp** et **iBoot** pour MacOSX et **Winload** pour Windows.

Source : wikimedia

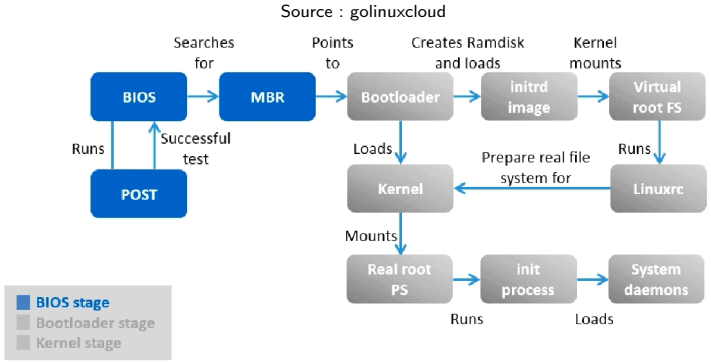


Legacy Boot



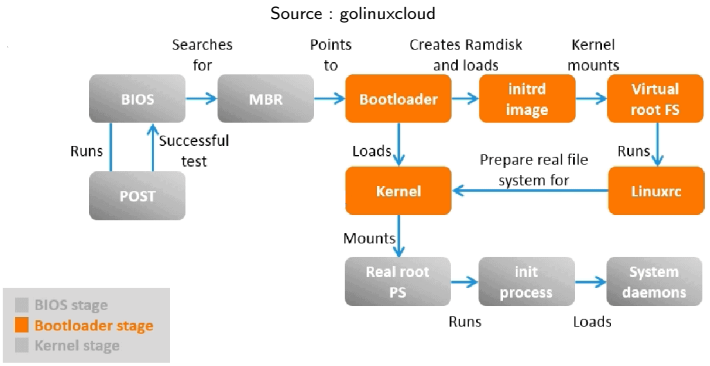
Procédure d'amorçage sous Linux

Source : golinuxcloud



Procédure d'amorçage sous Linux

Source : golinuxcloud

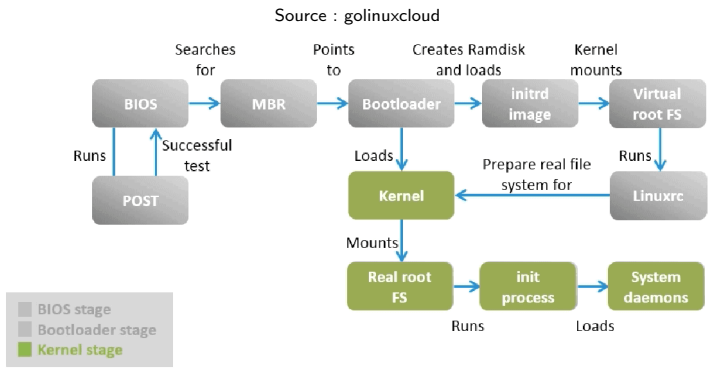


- Les éléments **kernel** et **initrd** sont des fichiers représentant respectivement le **code binaire du noyau** et un **système de fichiers minimal** chargés en RAM par GRUB.
- Le système de fichiers **initrd** est une **archive compressée** contenant l'ensemble des **programmes essentiels au démarrage** du système d'exploitation (*e.g.* montage des disques, initialisation du réseau).
- **initrd** n'est généralement utilisé que pour le démarrage du système, une **bascule** sur un système de fichiers stocké sur un **support de stockage** est effectuée par l'appel système **chroot** en toute fin d'initialisation.

Vincent Autefage Introduction à la Virtualisation 6/18

Procédure d'amorçage sous Linux

Source : golinuxcloud



- Le programme **init** est le **processus parent** de tous les processus qui tournent sur le système. Stopper **init** revient donc à **tuer l'ensemble des processus**.
- En règles générales, **init** joue également le rôle de **gestionnaire de services**, c'est à dire le programme chargé d'**orchestrer** les différents **services** et **démons** du système.
- Le programme **init** a plusieurs implémentations possibles dont les plus courantes sont **SystemD**, **SysV**, **OpenRC** ou encore **Runit**.

Processus

- Un processus est une **instance** d'un programme en **mémoire vive** dont les instructions sont exécutées sur le CPU (également sur le GPU dans certains cas).
- Chaque processus est identifié par un **PID** unique.
- Chaque processus est **attaché à son processus parent** (*i.e.* celui qui l'a instancié), **stopper le parent** implique ainsi l'**arrêt de tous ses processus fils**.
- Il est possible de **détacher** un processus de son parent pour l'**attacher à un autre** de plus haut niveau (généralement **init** qui possède le **PID 1**).
- Un processus peut **exécuter plusieurs fils d'instructions** en parallèle que l'on nomme **threads**.
- Tous les **threads** d'un même processus possède le même PID.
- À tout instant, un processus est dans un état particulier :
  - En cours d'exécution
  - En attente d'exécution
  - En sommeil
  - Zombie
- La commande **ps** permet de lister les processus en cours sur le système.
- La commande **kill** permet d'envoyer un signal à un processus permettant ainsi de le stopper plus (**SIGKILL**) ou moins (**SIGTERM**) violemment mais aussi de le mettre en sommeil (**SIGTSTOP**) ou le réveiller (**SIGCONT**).

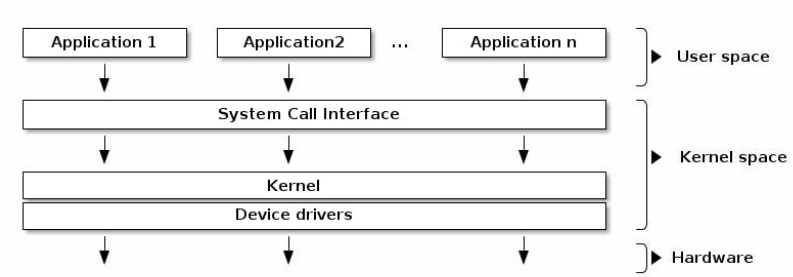
Vincent Autefage Introduction à la Virtualisation 7/18

Processus

- Lancez depuis le terminal le processus **gedit**.
- Depuis un autre terminal, tentez d'endormir le processus grâce à la commande **kill**, constatez sa mise en pause et réveillez le par la suite.
- Tuer le processus avec la même méthode.
- À quoi servent les commandes **fg** et **bg** ?
- Quelle est la différence entre **gedit** et **gedit &** ?
- Que se passe-t-il si vous fermez le terminal avec lequel vous avez lancé **gedit** alors que ce dernier est toujours en fonctionnement ?
- Refaite le même test mais en lançant **gedit** via la commande **setsid gedit**.
- Que pouvez-vous en déduire sur le rôle de la commande **setsid** ?
- À quoi servent les commandes **su** et **sudo** ?

Espaces noyau et utilisateur

Source : form3

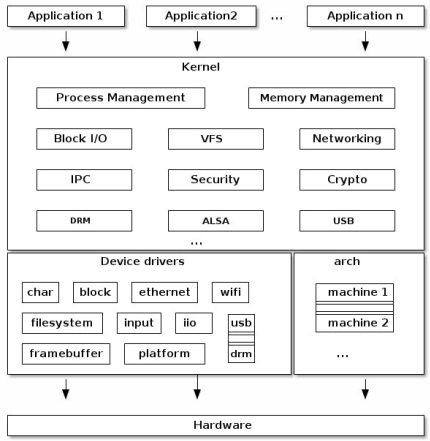


- Tout processus tourne dans ce que l'on nomme l'**espace utilisateur** (*User Space*) et dialogue avec le noyau au travers des **appels systèmes** (*Syscalls*).
- Les différents fils d'instructions qui s'exécutent au sein du noyau **n'ont pas de PID** dédié car ils ne représentent pas des processus à proprement parler.

Vincent Autefage Introduction à la Virtualisation 8/18

Espaces noyau et utilisateur

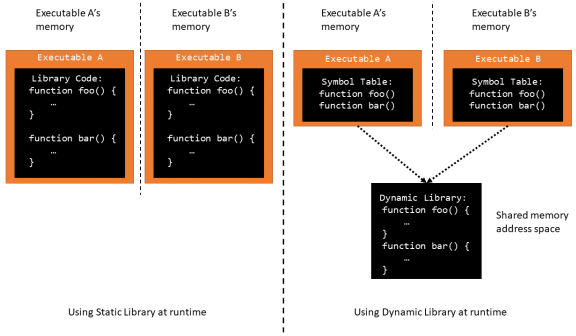
Source : form3



- Tout processus tourne dans ce que l'on nomme l'**espace utilisateur** (*User Space*) et dialogue avec le noyau au travers des **appels systèmes** (*Syscalls*).
- Les différents fils d'instructions qui s'exécutent au sein du noyau **n'ont pas de PID** dédié car ils ne représentent pas des processus à proprement parler.

Bibliothèques partagées

Source : medium / Domi Yan



- Il est possible de réutiliser des **collections de fonctions** compilées sous forme **statique** (*Static Library*) ou **partagée** / **dynamique** (*Shared Library*).
- Ces **bibliothèques** sont matérialisées par des fichiers : **.a** pour les statiques, **.so** (Linux), **.dll** (Windows), **.dylib** (Mac) pour les dynamiques.
- Une bibliothèque statique est **directement intégrée** dans le code de l'exécutable tandis que dans le cas d'une librairie dynamique, l'exécutable n'intègre que les **informations nécessaires à l'appel** des fonctions requises.
- Un exécutable **compilé dynamiquement** doit donc avoir accès **aux même versions des bibliothèques dynamiques** (compatibilité ascendante possible dans certains cas) auxquelles il fait référence.

Vincent Autefage Introduction à la Virtualisation 9/18

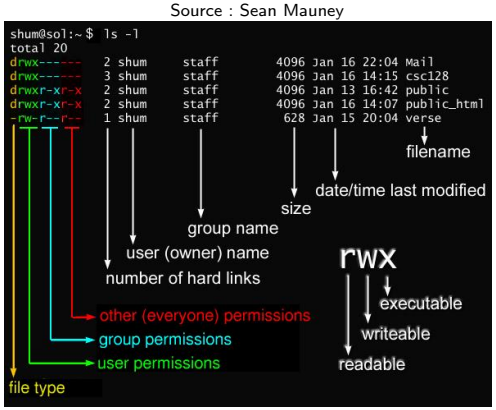
Bibliothèques partagées

```
--> ldd ~/iut-vms/vnet/nemu.d/rcd/build/nemo.Linux-x86_64

/usr/lib/x86_64-linux-gnu/libstdc++.so.6: version 'GLIBCXX_3.4.29' not found
/usr/lib/x86_64-linux-gnu/libstdc++.so.6: version 'CXXABI_1.3.13'
/lib/x86_64-linux-gnu/libc.so.6: version 'GLIBC_2.34' not found
/lib/x86_64-linux-gnu/libc.so.6: version 'GLIBC_2.32' not found

linux-vdso.so.1 (0x00007ffc58fc8000)
libboost_thread.so.1.74 => /usr/lib/x86_64-linux-gnu/libboost_thread.so.1.74 (0x00007f38b9b54000)
libboost_chrono.so.1.74 => /usr/lib/x86_64-linux-gnu/libboost_chrono.so.1.74 (0x00007f38b9b48000)
libboost_regex.so.1.74 => /usr/lib/x86_64-linux-gnu/libboost_regex.so.1.74 (0x00007f38b9a2b000)
libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f38b985e000)
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f38b971a000)
libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f38b96fe000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f38b9529000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f38b9507000)
libicui18n.so.67 => /usr/lib/x86_64-linux-gnu/libicui18n.so.67 (0x00007f38b9201000)
libicuuc.so.67 => /usr/lib/x86_64-linux-gnu/libicuuc.so.67 (0x00007f38b9018000)
/lib64/ld-linux-x86-64.so.2 (0x00007f38b9d01000)
libicudata.so.67 => /usr/lib/x86_64-linux-gnu/libicudata.so.67 (0x00007f38b74fd000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f38b74f7000)
```

- Sous Linux, le système cherche les bibliothèques dynamiques dans les répertoires listés dans `/etc/ld.so.conf`.
- Il est possible d'ajouter dynamiquement d'autres répertoires grâce à la variable d'environnement **LD_LIBRARY_PATH**.

Utilisateurs et droits

Source : Sean Mauney



- Tout processus et fichier (et par extension répertoire) est attaché à un utilisateur que l'on identifie par un **UID** unique.
- Les fichiers sont également attachés à un groupe identifié par un **GID** unique.
- Chaque fichier est associé à des **permissions** en **lecture**, **écriture** et **exécution** (**ouverture** pour les répertoires) l'on appelle **modes**.
- La commande **chmod** permet de modifier les permissions d'accès.
- La commande **chown** permet de modifier le propriétaire (utilisateur et groupe).

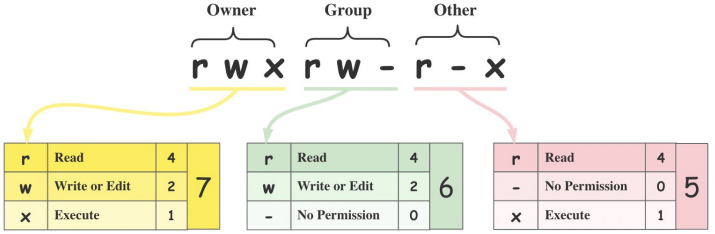
Vincent Autefage Introduction à la Virtualisation 10/18

Utilisateurs et droits

Linux File Permissions

blog.bytebytego.com

Binary	Octal	String Representation	Permissions
000	0 (0+0+0)	---	No Permission
001	1 (0+0+1)	--x	Execute
010	2 (0+2+0)	-w-	Write
011	3 (0+2+1)	-wx	Write + Execute
100	4 (4+0+0)	r--	Read
101	5 (4+0+1)	r-x	Read + Execute
110	6 (4+2+0)	rw-	Read + Write
111	7 (4+2+1)	rwx	Read + Write + Execute



Vincent Autefage Introduction à la Virtualisation 10/18

Utilisateurs et droits

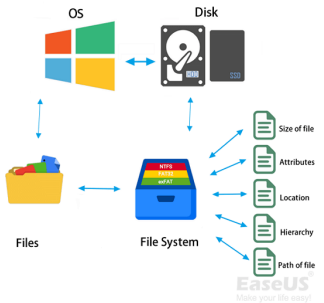
- Connectez vous en SSH sur la machine **info-biniou.iut.u-bordeaux.fr** sur le port **6666**.
- Créez un répertoire à votre nom dans `/tmp` accessible uniquement en ouverture pour les autres et créez quelques fichiers à l'intérieur de ce répertoire (certains en RW, d'autres en R et d'autres sans rien, vous pouvez tester plusieurs combinaisons).
- Demandez à un autre utilisateur de tenter d'accéder et de modifier votre répertoire ainsi que vos fichiers qui s'y trouvent.

Rappel

```
ssh -p <port> <user>@<host>
```


Système de fichiers

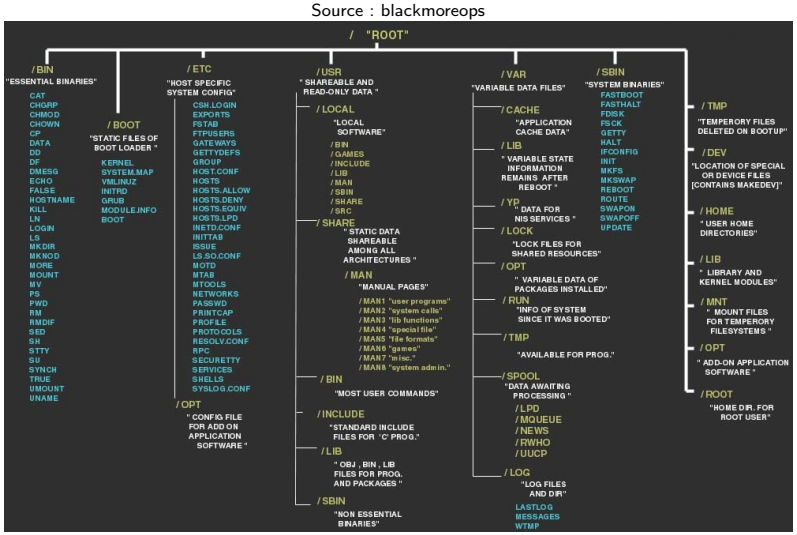
Source : easeus



- Un **système de fichiers** est une norme définissant l'organisation des données et des méta-données sur une support physique.
- Il existe un grand nombre de système de fichiers tels que **ext3**, **ext4**, **fat32**, **exfat**, **ntfs**, **hfs** ou encore **zfs**.
- Un support physique peut contenir **plusieurs systèmes de fichiers**, c'est ce que l'on nome communément le **partitionnement**.
- La **table des partitions** contient l'ensemble des informations permettant de **définir l'emplacement** physique **de chaque partition**.
- La **table des partitions** peut avoir plusieurs formats tels que **msdos** ou **gpt**.

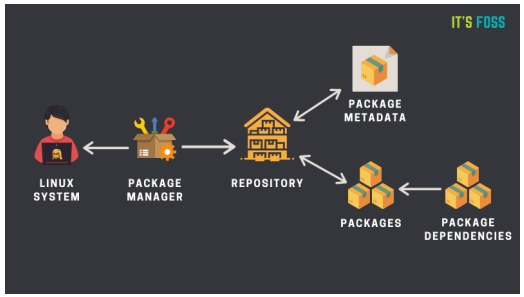
Organisation du système de fichiers sous Linux

Source : blackmoreops



- La **FHS** (**File Hierarchy System**) spécifie l'**organisation générale des fichiers** dans un système Linux.
- MacOS X s'est grandement inspiré de cette organisation, étant initialement basé sur BSD qui est lui même hérité de UNIX basé sur une FHS commune.
- Il peut exister quelques différences entre les distributions Linux mais la FHS est globalement très similaire.

Gestionnaire de paquets



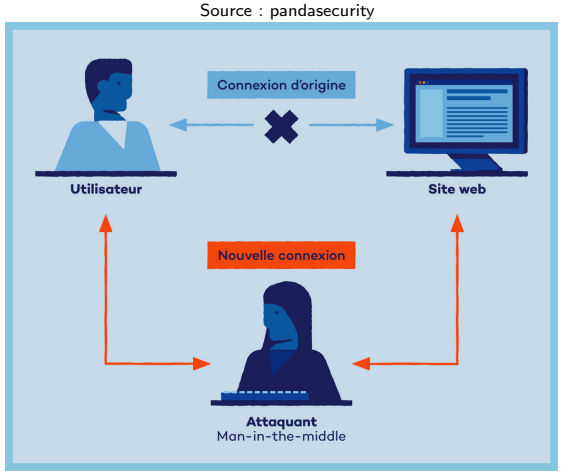
- Pour la majorité des distributions Linux, un **gestionnaire de paquets** (*Package Manager*) permet l'**installation**, la **mise à jour** et la **suppression** des différents logiciels ainsi que de leurs dépendances.
- Un **paquet** est matérialisé par une **archive compressée** qui peut être de différents formats (*e.g.* **deb**, **rpm**, **apk**).
- Une **dépendance** est un paquet nécessaire au fonctionnement d'un autre paquet.
- L'installation d'un paquet implique l'installation de ses dépendances ; par réciproque, la suppression d'un paquet implique la suppression de tous les paquets qui en sont dépendants.
- Le gestionnaire de paquets récupère les paquets depuis un **serveur web** que l'on appelle un **dépôt** (*Repository*).

Gestionnaire de paquets

- Trouvez les **formats de paquets** supportés par les distributions Linux suivantes : **Arch**, **Alpine**, **Debian**, **Fedora**, **OpenSuse**, **RedHat** et **Ubuntu**.
- Trouvez les **gestionnaires de paquets en ligne de commande** utilisés par ces différentes distributions.
- Existe-t-il des gestionnaires de paquets pour **Windows** et **Mac OS X** ? Si oui, lesquels ?

Transmission d'informations sensibles

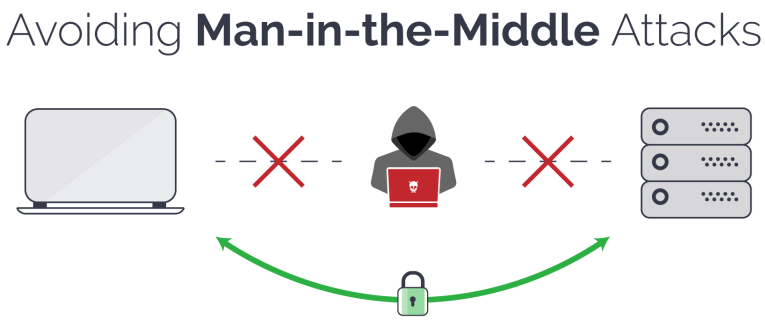
Source : pandasecurity



La technique de l'**homme du milieu** (*Man in the Middle*) consiste pour un fripon à **rediriger le trafic** entre deux machines vers une machine tiers qu'il contrôle afin d'**espionner** voir d'**altérer** les flux échangés.

Transmission d'informations sensibles

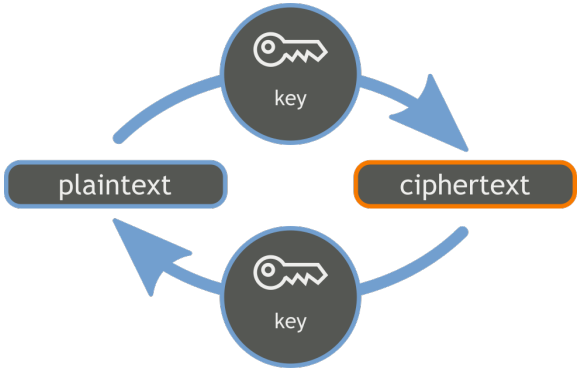
Source : invicti



La solution à ce problème consiste à **chiffrer et vérifier** la totalité du trafic afin d'empêcher toute écoute indiscrète ou modification fortuite.

Chiffrement, intégrité et signature

Source : wikimedia

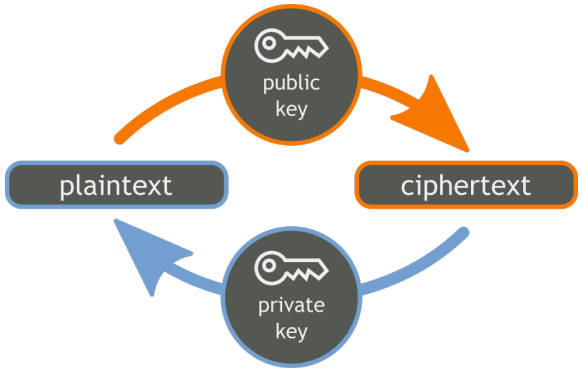


Chiffrement symétrique

Permet de **chiffrer** et de **déchiffrer** un message à l'aide d'une **même clé** dénommée **clé secrète**.
Exemples : **AES**, **ChaCha**.

Chiffrement, intégrité et signature

Source : wikimedia

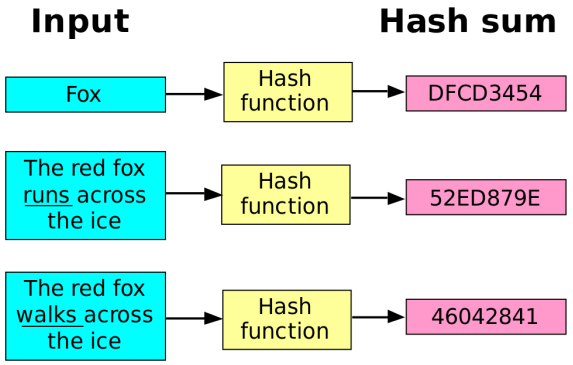


Chiffrement asymétrique

Permet de **chiffrer** un message à l'aide d'une **clé publique** et de **déchiffrer** le résultat à l'aide d'une **clé privée**.
Exemples : **RSA**, **ECC**

Chiffrement, intégrité et signature

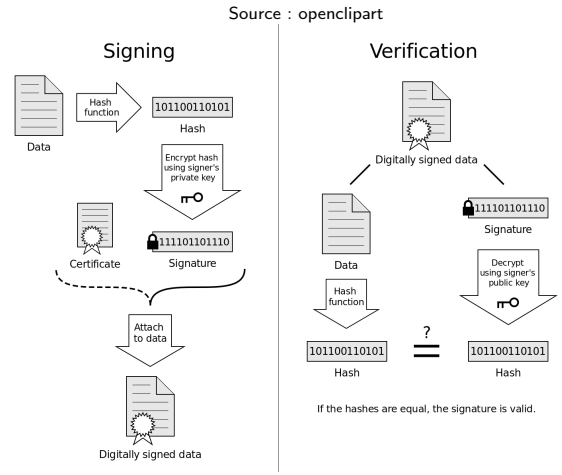
Source : wikimedia



Intégrité

Un message, même chiffré, **peut être modifié** de manière aléatoire par un attaquant qui se positionnerait sur le chemin emprunté par le message. Par conséquent, une **empreinte** (*Checksum*) est ajoutée au message chiffré. Une **fonction de hachage** permet d'associer à un élément un identifiant numérique **non réversible théoriquement unique** que l'on appelle **empreinte**. La probabilité qu'une même empreinte soit associée à 2 éléments distincts est appelée le **taux de collision**.
Exemples : **MD5**, **SHA**, **BLAKE**

Chiffrement, intégrité et signature



Signature

L'authentification consiste à **s'assurer de l'identité de son correspondant** par un moyen cryptographique appelé la **signature**. Une signature consiste à **chiffrer l'empreinte** d'un message avec une clé privée. La clé privée utilisée appartient à une entité de confiance que l'on appelle **autorité de certification**.

Le protocole SSL/TLS

- Le protocole **TLS**, anciennement **SSL**, utilise le **chiffrement asymétrique** pour sécuriser les échanges non seulement le web (*i.e.* HTTPS) mais également sur pléthore d'autres services (*e.g.* VPN, carte à puce).
- TLS utilise des **certificats** au format **x509** qui contiennent une **clé publique** ainsi que divers informations dont un **Common Name** et une **durée de validité**.
- Le **Common Name** est le **nom de domaine** d'accès au service.
- Un certificat TLS est **signé** par une **autorité de certification** attestant ainsi de la validité des informations qui s'y trouvent.
- Lors de la connexion à un service avec TLS, l'**ensemble des informations sont vérifiées** ainsi que la **reconnaissance de l'autorité de certification** grâce à sa clé publique.
- Un **manquement** ou une **anomalie** dans cette vérification entraîne l'**échec de la connexion** (sauf exception explicite).

Vincent Autefage Introduction à la Virtualisation 16/18

Le protocole SSL/TLS

- Le protocole **TLS**, anciennement **SSL**, utilise le **chiffrement asymétrique** pour sécuriser les échanges non seulement le web (*i.e.* HTTPS) mais également sur pléthore d'autres services (*e.g.* VPN, carte à puce).
- TLS utilise des **certificats** au format **x509** qui contiennent une **clé publique** ainsi que divers informations dont un **Common Name** et une **durée de validité**.
- Le **Common Name** est le **nom de domaine** d'accès au service.
- Un certificat TLS est **signé** par une **autorité de certification** attestant ainsi de la validité des informations qui s'y trouvent.
- Lors de la connexion à un service avec TLS, l'**ensemble des informations sont vérifiées** ainsi que la **reconnaissance de l'autorité de certification** grâce à sa clé publique.
- Un **manquement** ou une **anomalie** dans cette vérification entraîne l'**échec de la connexion** (sauf exception explicite).

Fabrication d'un certificat TLS/RSA auto-signé d'une durée de vie d'un an

```
openssl req -new -newkey rsa:4096 -x509 -sha256 -days 365 \
  -nodes -out <cert file> -keyout <key file>
```


Accès Moodle

R4A08 - Virtualisation

Groupe A

BUT_INFO_R4A08_S4**A**

Groupe B

BUT_INFO_R4A08_S4**B**

Groupe C

BUT_INFO_R4A08_S4**C**

Questions ?